Отчет по лабораторной работе № 6

“Исследование перегрузки операторов”

по дисциплине

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Выполнил студент группы ИВТ/б-22о

Горбенко К.Н.

Проверил:

Тимофеев И.С.

1. **Цель работы:** Исследование назначения и средств создания перегруженных операторов при написании объектно-ориентированных программ.
2. **Задание на работу:** создать класс Rectangle (хранит стороны a и b). Перегрузить операторы:
   1. ! как унарный метод класса, проверяющий, является ли прямоугольник квадратом;
   2. - как унарную функцию определения разности между длинами сторон;
   3. == как бинарный метод класса, сравнивающий два прямоугольника на равенство площадей;
   4. + как бинаную дружественную функцию нахождения общей площади фигур.
3. **Программа на языке С#.**

Класс Rectangle:

|  |
| --- |
| public class Rectangle  {  public double SideA { get; private set; }   public double SideB { get; private set; }   public Rectangle(double a, double b)  {  SideA = a;  SideB = b;  }   public Rectangle()  {  SideA = 0;  SideB = 0;  }   public static bool operator !(Rectangle rectangle)  {  return rectangle.SideA == rectangle.SideB;  }   public static double operator - (Rectangle rectangle)  {  return rectangle.SideA - rectangle.SideB;  }   public static double operator + (Rectangle rect1, Rectangle rect2)  {  var firstRectangleArea = rect1.SideA \* rect1.SideB;  var secondRectangleArea = rect2.SideA \* rect2.SideB;   return firstRectangleArea + secondRectangleArea;  }   public static bool operator == (Rectangle rect1, Rectangle rect2)  {  var firstRectangleArea = rect1.SideA \* rect1.SideB;  var secondRectangleArea = rect2.SideA \* rect2.SideB;   return firstRectangleArea == secondRectangleArea;  }   public static bool operator != (Rectangle rect1, Rectangle rect2)  {  var firstRectangleArea = rect1.SideA \* rect1.SideB;  var secondRectangleArea = rect2.SideA \* rect2.SideB;   return firstRectangleArea != secondRectangleArea;  }   public override bool Equals(object obj)  {  var rectangle = obj as Rectangle;  if (rectangle == null) throw new ArgumentNullException();   var firstRectangleArea = SideA \* rectangle.SideB;  var secondRectangleArea = SideA \* rectangle.SideB;   return firstRectangleArea == secondRectangleArea;  }   public override string ToString()  {  return $"A={SideA};B={SideB}";  }   public override int GetHashCode()  {  var hashCode = 999451561;  hashCode = hashCode \* -1521134295 + SideA.GetHashCode();  hashCode = hashCode \* -1521134295 + SideB.GetHashCode();  return hashCode;  }  } |

Класс Program:

|  |
| --- |
| class Program  {  static void Main()  {  var rect1 = new Rectangle(5, 5);  var rect2 = new Rectangle(10, 5);   WriteLine($"!rect1:{!rect1}");  WriteLine($"!rect2:{!rect2}");  WriteLine($"-rect1:{-rect1}");  WriteLine($"-rect2:{-rect2}");  WriteLine($"rect1==rect2:{rect1 == rect2}");  WriteLine($"rect1+rect2:{rect1 + rect2}");  ReadLine();  }  } |

1. **Результат работы программы:**

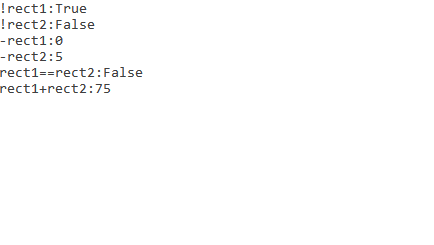
****

Рис. 1 - Результат работы программы

1. **Вывод:** в ходе лабораторной работы была изучена перегрузка операторов языка C#. Перегружающие методы обязательно должны быть статическими. В параметрах метода должен быть один параметр в случае, когда перегружаемая операция - унарная, и два параметра в случае, когда операция бинарная.